REMARKS

Claims 1-15 are currently pending in the application. By this amendment, claims 1-10 are amended and claims 11-15 are added for the Examiner's consideration. Support for the amendment(s) and added claims 11-15 is provided in at least Figures 3 and 6, and at page(s) 5-7 and 10-11 of the present specification. No new matter is added. Reconsideration of the rejected claims in view of the above amendments and the following remarks is respectfully requested.

35 U.S.C. §112 Rejection

Claim 10 was rejected under 35 U.S.C. §112, 2nd paragraph. This rejection is respectfully traversed.

By this Amendment, claim 10 is amended to provide proper antecedent basis for the term "interconnection logic means". Accordingly, claim 10 is in allowable condition and the rejection should be withdrawn.

35 U.S.C. §102 Rejection

Claim 1 was rejected under 35 U.S.C. §102(b) for being anticipated by U. S. Patent No. 5,430,850 issued to Papadopoulos, *et al.* ("Papadopoulos"). This rejection is respectfully traversed.

The invention is directed to hardware devices for processing the tasks of an algorithm in parallel, and dividing an algorithm into tasks where each task runs in its own task unit repetitively (in embodiments). With the algorithm divided as such, various processes and decisions may each run separately on a processor assigned to that task in order to speed up the processing of the algorithm.

The invention allows the processors or task units to be dedicated to specific operations or tasks. With such a configuration, the algorithm is subdivided into tasks and each task is directed to only a specific task unit. That is, no one task unit will receive all of the tasks or a same piece

of the task as another task unit for executing an action. By dividing the algorithm to allow any particular task unit to receive only a portion of the algorithm in the form of a specific task or action, a task unit may never receive the majority of tasks of the algorithm. Furthermore, the task units rely on the interconnection logic means for the routing of actions between source and destination task units. Thus, the system of the invention dedicates certain tasks to each task unit and allows for communication of actions between the task units to allow execution of particular tasks in the task unit based on a requested action. Benefits of the invention include simultaneously running the same task to improve efficiency and processing speed.

The advantages of the invention are accomplished by a hardware device for concurrently processing a plurality of tasks associated with an algorithm, which includes a plurality of task units for processing data, making decisions and/or processing data and making decisions. A task interconnection logic means interconnects the task units for communicating actions from a source task unit to a destination task unit. The task units also include a processor for executing the steps of the associated task in response to a received request action and runs the steps of the actions received from other task units. A status manager is provided for handling actions from source task units and building actions to be sent to destination task units.

In embodiments, the processor runs the steps of the actions received from other task units. The status manager manages the sending and receiving operations of a task unit based on the contents of the task unit and configuration registers. Accordingly, the status manager builds commands to be sent to other tasks units using a control field and a data field of the current task in conjunction with the contents of the two configuration registers. The control and data field of the task received by a status manager causes the status manager to load data into the processor which then processes the data. The status manager receives the task with the new data in the control and data fields when the process is completed by the processor, and builds a command using such data and the configuration registers to be sent to another task unit.

The task interconnection logic means serves to receive a task or action from the status manager of a task unit and analyzes at least a portion of the task to determine the appropriate

destination task unit. Thus, each task unit may be configured to repeatedly run only a particular task, and the task interconnection logic means guarantees that any particular task unit receives only those tasks for which it is configured to run.

In contrast, Papadopolous does not show all of the features of the presently claimed invention. In fact, after careful review of Papadopolous, it is clear that Papadopolous would teach away from the claimed invention. For example, Papadopolous shows providing the entire code or algorithm to each and every processor. This, according to Papadopolous, allows any action to run on any processor resulting in little down time when transferring data to a processor which already includes all of the code. The invention, though, is specific to running only certain tasks on certain task units.

More specifically, Papadopolous is directed to a multi-processor system having a plurality of processing nodes, where each node processes multiple threads of computation. The configuration of Papadopolous reduces processor idle time during data fetch operations of each processor of a multi-processor system by allowing the processors to run other computational threads while waiting for data. To accomplish this function, Papadopolous allows each node or processor to hold all steps of an algorithm, or dynamically receive steps of an algorithm, and start a new thread of steps while waiting for data to arrive for a previously running thread. As such, Papadopolous is configured to send all tasks to all processors. In contrast, the claimed invention selectively routes particular tasks to particular processors. That is, the invention does not have an entire algorithm loaded to each task unit for processing any action. Instead, only a distinct and separate portion of the algorithm is loaded to each task unit to execute an associate action and to allow multiple tasks to run concurrently to speed computation.

Additionally, Papadopolous allows a processor to run any task when the necessary data arrives at that processor. In contrast to the claimed invention which dedicates certain processors to certain tasks.

As to the Examiner's specific remarks, the Examiner asserts that Papadopolous shows task interconnection logic means interconnecting the task units for communicating actions from a

source task unit to a destination task unit at col. 4, lines 64-66, col. 5, lines 40-45, and col. 20, lines 8-23. But, Papadopolous actually shows a synchronization coprocessor which indicates to a data processor when a new thread of computation may be performed. In operation, the synchronization coprocessor processes start messages to store data to be used in computational threads. Additionally, the synchronization coprocessor determines when all data required for a thread to run has been received at the processor and the thread of computation may be initiated. Thus, the synchronization coprocessor of Papadopolous only operates on data, and does not communicate actions from a source task unit to a destination task unit, as in the claimed invention.

More particularly, the synchronization coprocessor processes start messages from the same and other nodes of the system. Such a synchronization coprocessor is required because it allows data to be received by a node and linked to the appropriate computational thread at the node. In operation, the synchronization coprocessor processes the start messages from other nodes and stores values thereof as operands for threads of computation. The synchronization coprocessor also determines when all data required for a thread of computation has been received. Additionally, the synchronization coprocessor provides an indication to the data processor that a thread of computation may be initiated because all the data needed for the computation has arrived at the node.

However, the synchronization coprocessor of Papadopolous does not send or receive any action or computational thread. This is because synchronization coprocessor handles only data. The synchronization coprocessor receives data and links it with the appropriate thread residing in the node. Papadopolous is specifically configured so that each processor can run any portion of the computational thread to avoid processor idle time during data fetch operations, whereas the invention relies on task units dedicated to particular actions.

The Examiner also asserts that Papadopolous shows a status manager for handling actions from source task units and building actions to be sent to destination task units at col. 26, line 54-60. But, Papadopolous actually shows a RMem processor which receives messages destined for a

current node, and either sends a message back to the requesting processor or elicits a trap on the current processor for handling conditions beyond its capability. Thus, the RMem processor of Papadopolous only communicates with a source processor or its own processor. The limited communication options of the RMem processor is in contrast to the status manager of the claimed invention which handles actions from multiple source task units and builds actions to be sent to multiple destination task units, in addition to communicating with its own task unit.

Consequently, Papadopolous fails to show a task interconnection logic means interconnecting task units for communicating actions from a source task unit to a destination task unit, and a status manager for handling actions from source task units and building actions to be sent to destination task units, as set forth in claim 1. Consequently, claim 1 is in allowable condition.

Accordingly, Applicants respectfully request that the rejection over claim 1 be withdrawn.

35 U.S.C. §103 Rejection

Claims 2-9 were rejected under 35 U.S.C. §103(a) for being unpatentable over Papadopolous in view of Official Notice. Claim 10 was rejected under 35 U.S.C. §103(a) for being unpatentable over Papadopolous in view of U. S. Patent No. 5,321,842 issued to Fairfield, *et al.* ("Fairfield"). These rejections are respectfully traversed.

I. Official Notice

Documentation is typically required to support Official Notice: "Ordinarily, there must be some form of evidence in the record to support an assertion of common knowledge. *In re Lee*, 277 F.3d 1338, 1344-45, 61 USPQ2d 1430, 1434-35 (Fed. Cir. 2002)." MPEP 2144.03(B). Undocumented Official Notice must include specific factual support: "The examiner must provide specific factual findings predicated on sound technical and scientific reasoning to support his or her conclusion of common knowledge. *In re Soli*, 317 F.2d 941, 946, 137 USPQ

797, 801 (CCPA 1943)." *Id.* Official Notice may be traversed by pointing out errors in the examiner's action, including why the fact is not common knowledge. 37 C.F.R. 1.111(b); MPEP 2144.03(C). If Official Notice is adequately traversed by the applicant, the examiner must provide appropriate documentation or withdraw the Official Notice. 37 C.F.R. 1.104(c)(2); MPEP at *id.*

The Examiner notes that Papadopolous does not specifically teach the actions to include KILL used to cancel the task associated with the destination task unit and VALID used to confirm that a task associated with said destination task unit corresponds to a decision included in said task. The Examiner then takes undocumented Official Notice that it would been obvious to implement Papadopolous' system with commands such as KILL and VALID to perform administrative actions. Applicants traverse the undocumented Official Notice, and note that no factual basis is provided to support the assertion that it is common knowledge that administrative commands which function in system of one particular design would properly function in a second system of a different particular design. Accordingly, Applicant respectfully traverses the Examiner's assertion that it would have been obvious to add different commands to perform desired actions according to administrating needs (e.g., well known in the art) and pursuant to M.P.E.P. §2144.03 requests the Examiner to (i) withdraw the rejection or (ii) supply evidence, e.g., a prior art reference, supporting the asserted "well known" feature. Accordingly, Applicants request the Examiner to provide a factual basis for documentation indicating that the Papadopolous system would properly function using the KILL and VALID commands as defined in the Applicants' specification.

II. Papadopolous and Official Notice

Even if proper Official Notice is provided, the combination of Papadopolous and Official Notice neither disclose or suggest all the claimed features of the invention. Applicants note that a §103 rejection requires the Examiner to first establish a prima facie case of obviousness: "The examiner bears the initial burden of factually supporting any prima facie conclusion of

obviousness. If the examiner does not produce a prima facie case, the applicant is under no obligation to submit evidence of nonobviousness." M.P.E.P. § 2142. The Court of Appeals for the Federal Circuit has set forth three elements which must be shown for prima facie obviousness:

> To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

Papadopolous does not set forth actions communicated from a source task unit to a destination task unit, as set forth in claim 2 because Papadopolous does not communicate actions between task units. The Official Notice that "commands such as KILL and VALID to perform administrative actions" fails to provide the missing communication of actions between task units lacking in Papadopolous, and thus fails to cure the deficiencies of the reference.

Regarding claim 3, Papadopolous and the Official Notice neither disclose or suggest a status manager which activates a processor for processing the steps of an associated task with said destination task unit when the action received from a source task unit is START. Papadopolous does not have a destination task unit which receives an action. Papadopoulos also fails to show a source task unit which sends an action. The Official Notice fails to cure these deficiencies of Papadopolous.

For the reasons set forth above, claims 2 and 3 are in allowable condition. Claims 4-9 are allowable at least for the reasons set forth above with respect to claims 2 and 3, from which they depend, as well as for their added features. Applicants respectfully request that the rejection of claims 2-9 be withdrawn.

III. Papadopolous and Fairfield

The Examiner cites Fairfield for showing using three-state drivers to help employ feedback to the processor in Papadopolous' system at col 2, lns 4-16. However, Fairfield actually shows multiple processors accessing a memory through a shared bus, whereby interference between the processors is prevented by a three-state driver in each processor which outputs a "logic active," "logic inactive," or on "off" signal. The three-state driver of each processor prevents conflicts between its processor and the other processors by producing the appropriate signal when its processor needs to access the bus or does not need to access the bus. When a three-state driver of a processor signals it is accessing the boss, the other processors do not attempt to access the bus at the same time.

However, Fairfield shows multiple three-state drivers associated with a single bus, and each driver being associated with multiple signals. Fairfield also shows each driver operating on a single bus. As such, the three-state driver of Fairfield is not a driver associated with one of said tasks as input task and a number of buses is equal to the number of tasks as output tasks, and one of the buses being selected by the driver corresponding to an input task, as set forth in claim 10. Consequently, Fairfield does not cure the deficiencies of Papadopoulos and claim 10 is in allowable condition.

Accordingly, Applicants respectfully request that the rejection over claim 10 be withdrawn.

Other Matters

Claims 1-10 have been amended for informalities. Prompt examination and allowance in due course of all pending claims is respectfully requested.

New Claims

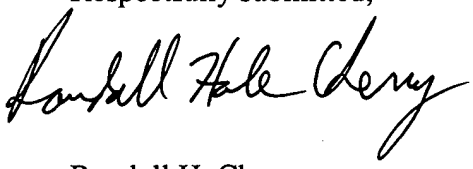
By this amendment, new claims 11-15 are added. Claim 11 depends from allowable claim 1 and should be in allowable condition. Claim 12 sets forth, amongst other allowable

features, an interconnection logic means for routing actions from a source task unit to a destination task unit, and thus should be in allowable condition. Claims 13-15 depend from claim 12, and thus should be in allowable condition. Prompt examination and allowance in due course of new claims 11-15 is respectfully requested.

CONCLUSION

In view of the foregoing amendments and remarks, Applicants submit that all of the claims are patentably distinct from the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue. The Examiner is invited to contact the undersigned at the telephone number listed below, if needed. Applicant hereby make a written conditional petition for extension of time, if required. Please charge any deficiencies in fees and credit any overpayment of fees to Attorney's Deposit Account No. 09-0457.

Respectfully submitted,



Randall H. Cherry
Registration No. 51,556

Andrew M. Calderon
Registration No. 38,093

McGuireWoods, LLP
Suite 1800
1750 Tysons Blvd.
McLean, VA 22102
(703) 712-5426

/comm.//#389546